

PDF-Reuse Perl Module

Description

This module could be used when you want to mass produce similar (but not identical) PDF documents and reuse templates, JavaScripts and some other components. It is functional to be fast, and to give your programs capacity to produce many pages per second and very big PDF documents if necessary.

Functions

- Mandatory Functions
 - [prFile - define output](#)
 - [prEnd - end/flush buffers](#)
- Optional Functions
 - [prAdd - add ``low level" instructions](#)
 - [prBookmark - define bookmarks](#)
 - [prCompress - compress/zip added streams](#)
 - [prDoc - include pages from a document](#)
 - [prDocDir - set directory for produced documents](#)
 - [prDocForm - use an interactive page as a form](#)
 - [prExtract - extract an object group](#)
 - [prField - assign a value to an interactive field](#)
 - [prFont - set current font](#)
 - [prFontSize - set current font size](#)
 - [prForm - use a page from an old document as a form/background](#)
 - [prGetLogBuffer - get the log buffer](#)
 - [prGraphState - define a graphic state parameter dictionary](#)
 - [prImage - reuse an image from an old PDF document](#)
 - [prInit - add JavaScript to be executed at initiation](#)
 - [prInitVars - initiate global variables and internal tables](#)
 - [prJpeg - import a jpeg-image](#)
 - [prJs - add JavaScript](#)
 - [prLink - add a hyperlink](#)
 - [prLog - add a string to the log](#)
 - [prLogDir - set directory for the log](#)
 - [prMbox - define the format \(MediaBox\) of the current page.](#)
 - [prPage - insert a page break](#)
 - [prStrWidth - calculate the string width](#)
 - [prText - add a text-string](#)
 - [prTouchUp - make changes and reuse more difficult](#)

Note:

If the first page is the first interactive component (`prDoc()` or `prDocForm()`) the interactive functions are kept and also merged with JavaScripts you have added, if any. But, if you specify a first page different than 1 or a last page, no JavaScript are extracted from the document, because then there is a risk that an included JavaScript function might refer to something not included.

Only supports one stream...so cannot add multiple interactive documents, then populate form fields in each of those documents..which is a limitation. But can use documents as templates, then using `prText()`, position text fields on the form using x,y coordinates. The `prText()` function may still be a hassle, but now we do not have to re-draw the form using the `PDF::Create` module.

Example Creating File (directory must have write permissions)

```
use CGI qw(:standard);
use CGI::Carp qw(fatalToBrowser);
use PDF::Reuse;

#####
#### creates copy of PDF and populates form fields with data #####
#### cory.fischer@ded.mo.gov 08.04.04 #####
#####

$begindate = "07/01/04";
$enddate = "07/31/04";

prDocDir('c:/inetpub/wwwroot/DMD/cgi-bin'); ## document directory
prFile('myfile.pdf'); ## file to be created, required
prField('txtOrgName', 'Department of Economic Development'); ## Form Field, value
prField('txtProjNum', 'J 01110101012');
prField('txtFundBegin', $begindate);
prField('txtFundEnd', $enddate);

prDocForm('c:/inetpub/wwwroot/DMD/cgi-bin/2004QuarterlyReport.pdf'); # document to reuse (copy)
prEnd(); # required

## print out pretty message, could do a re-direct if wanted ##
print "Content-type: text/html\n\n";
print qq(<html><head><title>PDF Created Example</title></head><body>);
print qq(<h2 align="center">Your file has been successfully created!<br><br>Click below to view your
created PDF.</b><br><br>);
print qq(<a href="/dmd/cgi-bin/myfile.pdf">MYFILE.PDF</a></h2>);
print qq(</body></html>);
```

Pros of using this method

1. Printable portable dynamic-generated cross-browser web document
2. Good for internal use if needing to save dynamically generated PDF documents
3. Can save file to server by dynamically creating a filename to save as (no duplicates)
4. User can save a copy (local pc or on network)

Cons of this method

1. Must grant write access to web directory

Example using STDOUT (no permission changes needed, just Adobe Reader)

```
use CGI qw(:standard);
use CGI::Carp qw(fatalToBrowser);
use PDF::Reuse;

#####
#### creates PDF and populates form fields with data #####
#### sends to STDOUT cory.fischer@ded.mo.gov 08.04.04 #####
####

$begindate = "07/01/04";
$enddate = "07/31/04";

prDocDir('c:/inetpub/wwwroot/DMD/cgi-bin'); ## document directory

prInitVars(); # globalizes variables

$|=1; # autoFlush

print STDOUT "Content-Type: application/pdf\n\n";

prFile(""); ## send to STDOUT, required

prField('txtOrgName', 'Department of Economic Development'); ## Form Field, value
prField('txtProjNum', 'J 01110101012');
prField('txtFundBegin', $begindate);
prField('txtFundEnd', $enddate);

prDoc('c:/inetpub/wwwroot/DMD/cgi-bin/2004QuarterlyReport.pdf'); # document to reuse (copy)
prEnd(); ## required
```

Pros of using this method

1. Printable portable dynamic-generated cross-browser web document to STDOUT
2. Works great for public Web sites
3. No directory permission changes because no pdf file created on server
4. User can save a copy (local pc or on network)

Cons of this method

1. Does not save to server
2. Reader versioning issues?

For more information...

Documentation

- <http://search.cpan.org/dist/PDF-Reuse/Reuse.pm> - Cpan Documentation
- <http://perladvent.org/2003/18th/>
- Google Search on “perl pdf-reuse”

Documentation/Install

- <http://ftp.sh.cvut.cz/MIRRORS/CPAN/modules/by-module/PDF/PDF-Reuse-0.24.readme>

If you cannot find what you are looking for, contact me and I will provide the best assistance I can.

Cory Fischer
DED/MIS
Cory.fischer@ded.mo.gov
522-4529